UNIVERSITY OF TOURS – HONG DUC UNIVERSITY
POLYTECH TOURS
--- 🕮 ---

REPORT

Video Capture optimization with latency estimation on TV Workstation

Xuan-Quang LE
xuanle.quang@etu.univ-tours.fr
lexuanquang1408@gmail.com
+33 626 752 778 (FR)

Mme. Tifenn RAULT
M. Mathieu DELALANDRE
Supervisors

Delivered on: 00/00/2024

Table of Contents

1. Introduction

This is a very important part of our studies in the eighth semester, within the framework of the "Graduation Thesis" program.

Under the guidance of researchers from Polytech Tours, Ms. Tifenn RAULT and Mr. Mathieu Delalandre, who serve as Supervisors, we took on the responsibility of clearly understanding their requirements and providing a solution that meets their expectations.

Our project focuses on enhancing the TV station system that was initiated in 2018 by Polytech Tours and the LIFAT institute. This advanced TV station, equipped with 8 capture cards, supports recording live TV programs. Its technical capabilities open up opportunities for new projects in data journalism, fact-checking, and social TV. However, the current system of the TV station is limited to recording only 8 programs simultaneously. To meet the fluctuating recording demands, scheduling algorithms have been implemented to optimize program recording. Therefore, we will develop a solution to optimize video recording with latency estimation on the TV Workstation.

This project is related to the video capture problem on TV Workstation [1, 2]. A TV workstation is a computer designed to capture the television signal with a specific hardware equipment and configuration (e.g. a multituner to decode the DTT signal, decoding video cards, large disk capacity for storage, high performance CPU for audio coding, etc.) [1]. Due to the scalability and storage issues, such a workstation can record a limited number of TV channel at a time (e.g. 8). This raises a problem of parallel machine scheduling (PMS) to capture the right video content within the entire DTT signal, as some tens channels can be delivered at the same time (e.g. a near 30 to 40 in France) [2]. It can be characterized as a Weighted Job Interval Selection Problem (WJISP) that could be addressed with parameterized algorithms such as the GREEDYα [3]. A public dataset has been published on that problem [2], as well a public implementation of the GREEDYα algorithm [4] and benchmarking tools for performance evaluation [5].

2. TV Workstation

2.1.The architecture of the TV Workstation

The station is equipped with two machines: one machine processes the real-time video stream and another machine captures the video. DELL 5820 computer processes 8 channels (HD, 30 FPS, 24h/day), with real-time audio / video encoding, control of tuners with IR sensors, internal / external storage of 38 + 80 TB :

- Video Processing Capability: The DELL 5820 processes 8 HD video channels at 30 FPS, ensuring smooth and high-quality video capture around the clock.
- Real-Time Encoding: With real-time audio and video encoding, the system effectively compresses and stores video data without compromising quality.
- Automated Tuner Control: Utilizing IR sensors, the system automates the switching and control of TV tuners, enhancing operational efficiency.
- Massive Storage: A combined total of 118 TB (38 TB internal + 80 TB external) ensures that the system can handle extensive video storage requirements.

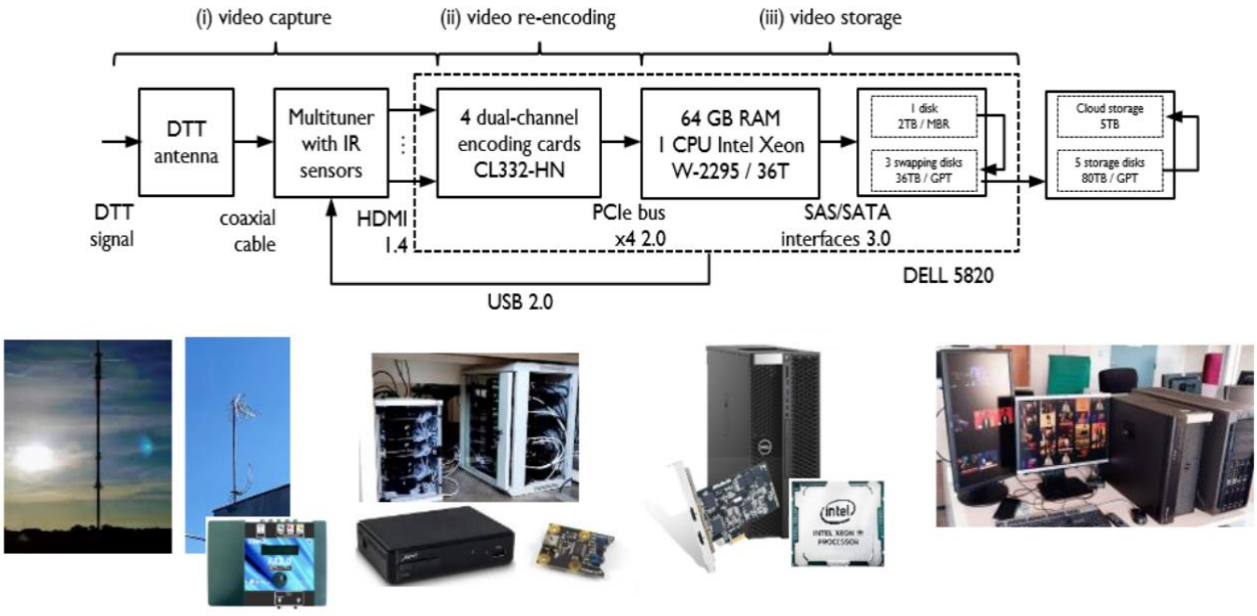
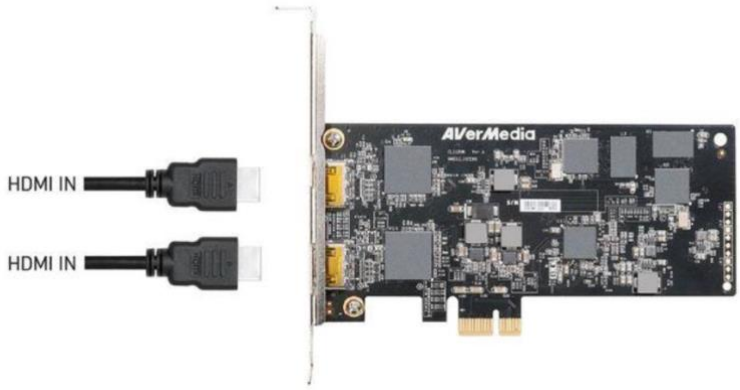


Figure 1: The architecture of the TV Workstation

Video recording is supported by the Avermedia CL332-HN hardware card embedded in the DELL T7610 machine and connected to a multi-controller. The multi-controller is composed of individual Astrell-brand controllers. The architecture of the CL332-HN/multituner card allows for capturing 8 video streams simultaneously in the station.



➢ *Figure 2: Avermedia CL332-HNPCle HD capture device:*

• Support Dual-Channel

• Full-HD capture (1920 x 1080 30fps)

• Hardware H.264 encoding

• Standard HDMI connector

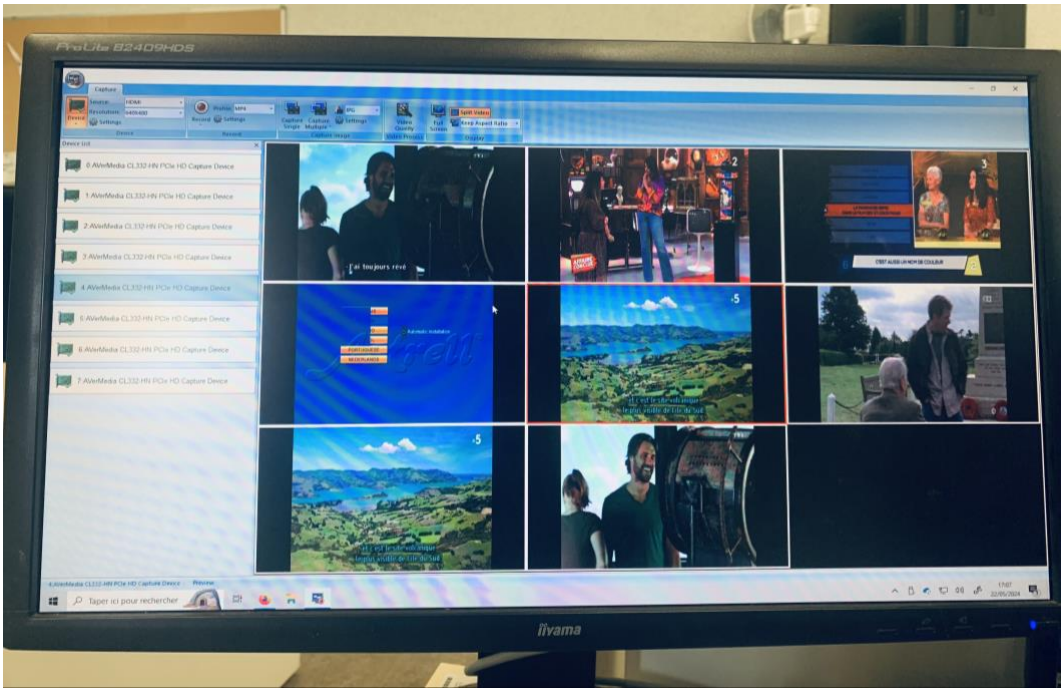


Figure 2.1: Avermedia CL332-HNPCle HD capture device

The Astrell controllers in the multi-controller are controlled via the PhidgetIR IR 1055 transmitter connected through a USB splitter. These controllers use the standard NEC protocol with the system code 0x08F6. This hardware configuration (Astrell controller/PhidgetIR IR 1055 transmitter) allows the television station to "attack" between 30 TNT channels.

2.2.Problems and solutions

The table below outlines the performance metrics of the DELL 5820 computer for video storage, categorized by different resolutions and corresponding audio/video data rates and monthly data usage:

Resolution		Audio/ video		Video Mbps	TB/ month		Audio Kbps	GB/ month
HD	1280x720	asyn		3	7.23		256	621
SD	720x576			1.6	3.89		160	384
Low	320x240			0.56	1.36		128	308

Figure 3: DELL 5820 computerperformance for video storage

These values provide a snapshot of the data throughput capabilities of the DELL 5820 computer across different video resolutions. The monthly data usage metrics illustrate its capacity to handle varying video streaming requirements efficiently. This information is crucial for understanding the suitability of the DELL 5820 for video storage and playback applications, ensuring optimal performance and resource management in multimedia environments

Modes	Details	Pro	Cons
Static and continue capture	• Each input of card is assigned to a single channel with a static assignment (no switch between the channels during the capture). • The capture cannot be made idle.	Easy to setup	• No break, require long captures and a huge storage • Cannot filter the duplicate data (twice daily shows) • Can capture only 8 different channels.
Dynamic capture	• Each input of card can be controlled to switch between channels, even made idle for a time between two captures.	• Require short captures with a limited storage • Can capture many different channels at one time (>8) • Can filter the duplicate data	More difficult to handle

Figure 4: we capture along in multiple channels with two modes

When capturing with multiple channels, it requires a lot of resources for storage. in addition, it also has a lot of Duplicate data, repeated data. That's the reason we try to developed solution with two level:

- An algorithm to optimize the scheduling
- A runtime for dynamic capture

3. Optimization of the selection of the programs to record

3.1. The algorithm to optimize the scheduling

The Kleinberg algorithm is a solution to the maximum weight non-overlapping interval scheduling problem. This problem requires finding a subset of non-overlapping interval such that the sum of their weights is maximized.

The TV workstation can record up to 8 TV programs simultaneously since it is equipped with 8 recording cards. However, the television programs offer is much more important since there are more than twenty channels. Therefore, we must select the programs to be recorded. In order to do that, we assume that each program is associated with a weight that represents its importance. We then use algorithms that optimize this selection, i.e that maximize the sum of the weights of the selected programs.

In what follows, an interval represents a specific time period during which a program is recorded. The optimization goal is to maximize resource utilization, minimize conflicts between intervals, and ensure a high-quality viewing experience.

Channel	Hashcode	Start Date	Stop Date	Weight
1	5d2c9463d885cd62	0	15	28
1	8df4ddf1335c5db2	20	40	27
1	fab07ab63e8d5bb3	40	50	73
1	b27ff3d07cdf7f3a	60	100	36

Figure 5: Extract from an input file for our tool

Name of file: “Instance_ByDay_2020_12_7.csv”

The input of our problem is the list of all the programs that will be broadcasted during one day. Each line in the file (see Figure 2) represents a television program, described by the channel, the show hashcode, the start and end times, and a weight indicating its importance (also called the value) Our aim is to choose a set of programs that maximizes the sum of the weights. This problem is known as the **Weighted Interval Scheduling Problem** (WISP).

The Kleinberg algorithm efficiently solves the Weighted Interval Scheduling Problem for one machine. It is a significant algorithm in the field of dynamic programming and is widely applied in various real-world applications, including TV workstation systems.

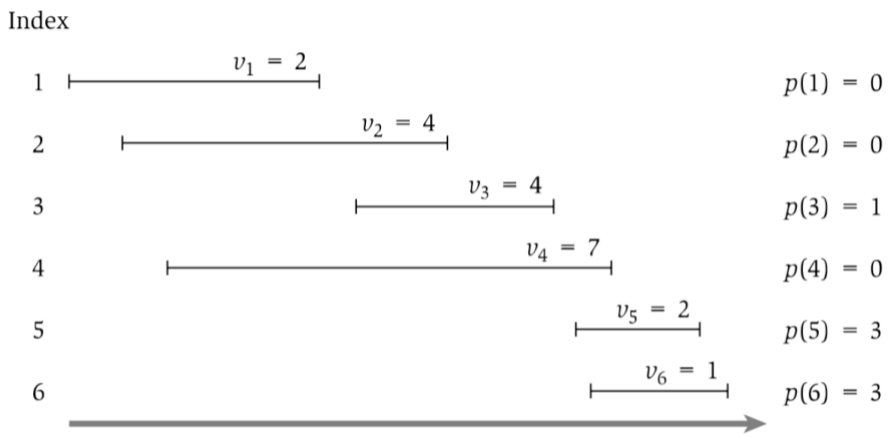


Figure 6: A simple instance of weighted interval scheduling.

In the program was developed this algorithm and adapted it to work for many machines.In fact, we have to run the algorithm once, then we have to remove the selected intervals, and then we re-run the algorithm a second time. And so forth, until we execute it a total of 8 times.

In TV workstations, applying the Kleinberg algorithm can yield good performance in managing broadcast schedules and recording television programs. By optimizing the program recording schedule, this algorithm can enhance the flexibility and efficiency of TV stations. This can lead to an improved viewing experience by providing continuous and high-quality television programming, while significantly reducing latency during schedule intervals. However, the specific performance of the algorithm in TV workstation environments may depend on various factors, including the size of input data, the complexity of the broadcast schedule, and the specific implementation approach of the algorithm, etc.

The Kleinberg algorithm is not the only algorithm that can be used to solve our problem. Indeed, in the literature, other ways have been proposed. For example, Erlebach and Spieksma introduced the Greedy alpha algorithm (reference: Interval selection: Applications, algorithms, and lower bounds 2003). The Greedy alpha algorithm has already been developed by a student of Polytech.

To evaluate the performance of the different algorithms, we will base our analysis on their output files (Excel CSV files).These output files must comply with the writing standards detailed below.

Figure 3 there is an example of an output file. It contains the list of the selected intervals.

This is an example of output files:

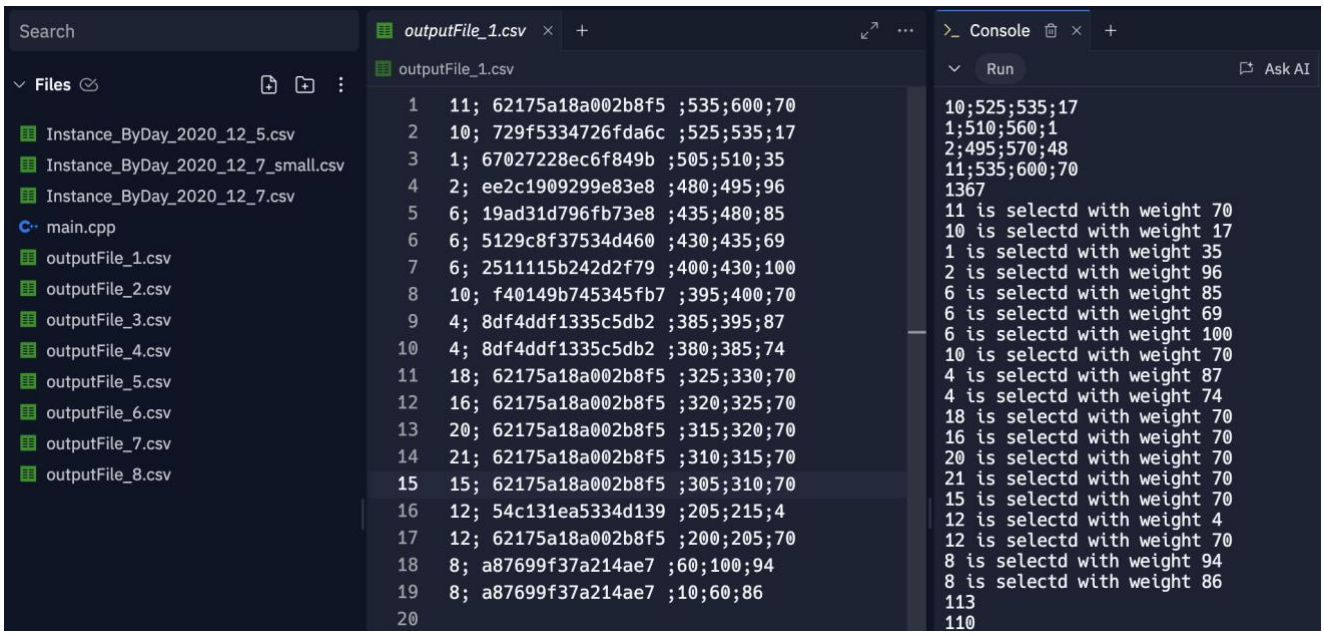


Figure 7: Extract from an output file The Kleinberg algorithm is run with different parameters

At Polytech, two students undertook the development of a comprehensive tool to evaluate and compare the performance of different algorithms. Our goal is to improve and optimize that algorithm.

2.2. Presentation and implementation.

Here we detail the different steps of the Kleinberg algorithm and its adaptation to our specific problem:

- **Step 1:** Sort intervals by end_time

In this step, the intervals are sorted by their end_time using the quicksort algorithm. This allows us to obtain a list of intervals sorted from earliest to latest.

Using the QuickSort function. With this function, it sorts the intervals based on their end time. This sorting process can help optimize the selection of TV programs by end time, making it easier to compute the P values and find the optimal solution for the problem.

The "Check overlap" function is used to determine whether two intervals overlap each other or not. Specifically, it checks if the end time of interval `a` comes before the start time

of interval `b`. If this condition is met, it means interval `a` extends into interval `b`, indicating that they overlap each other. This function only works if intervals ‘a’ and ‘b’ are already sorted.

- **Step 2:** Compute P values

The P values (also known as p values) are calculated for each interval. The P value of an interval is defined as the index of the interval that ends before the current interval without overlapping. This ensures that only a limited number of TV programs are recorded simultaneously, preventing the recording system from being overloaded.

We iterate through each interval and compute its corresponding P value, which represents the index of the interval that ends before the current interval without overlapping. Specifically, for each interval `i`, we compare its end time (`end_time[i]`) with the start time of all previous intervals (`start_time[j]` for `j` from 1 to `i-1`). If the end time of interval `i` is greater than or equal to the start time of interval `j`, then we update the P value of interval `i` to `j`. This ensures that the P value indicates the index of the latest non-overlapping interval before the current interval. If no such interval is found, the P value of interval `i` is set to 0, indicating that there is no interval that ends before the starting time of ‘i’.

- **Step 3:** Compute M values

In this step, an array M is used to store the optimal values for each interval j, which represents the optimal if we consider intervals from 1 to j. This is computed based on the P values calculated in the previous step and the weights of the programs.

The `M_compute_OPT` function computes the optimal solution for the given problem instance, considering the weights and dependencies of intervals. It uses dynamic programming to efficiently calculate the maximum weight of the selected intervals. This function takes into account the values of `p`, which represent the previous non-overlapping interval for each interval, and `v`, which represent the weights of the intervals. The function

recursively computes the maximum weight of the selected intervals, considering whether to include the current interval or skip it based on its weight and the weight of previously selected intervals.

- **Step 4:** Find Solution

Finally, after computing the P and M values, the optimal solution is searched for using these values. Specifically, starting from the last M value, the program searches for the selected TV programs by iterating through the computed P and M values, ensuring that they are non-overlapping and do not duplicate.

Selected intervals (`Find_Solution`):Once the optimal solution is computed, the `Find_Solution` function is invoked to identify the intervals that contribute to this optimal solution. It recursively traces back through the computed optimal solution to identify which intervals were selected.

Printing the selected intervals:As the `Find_Solution` function traverses through the optimal solution, it prints out the index of each selected interval along with its weight. This provides insight into which intervals are included in the optimal solution and their corresponding weights.

By printing the selected intervals, Step 4 helps validate the correctness of the algorithm and provides visibility into which programs are chosen for recording based on their importance (weight) and dependencies.

Adaptation of the Kleinberg Algorithm to our problem. Since the Kleinberg algorithm is optimal in the context of one machine, we need to adapt it to our application problem where we have 8 machines (cards)

- **Step 5:**

We remove the selected intervals : they will be recorded on one card. Then, we re-run the algorithm. The new set of selected intervals will be recorded on the second card. We suppress those selected intervals, and re-run the Kleinberg algorithm, and so on until we have

8 sets of selected intervals : one for each card .

Through this iterative approach, we ensure efficient utilization of all available resources and optimize the recording process across multiple machines.

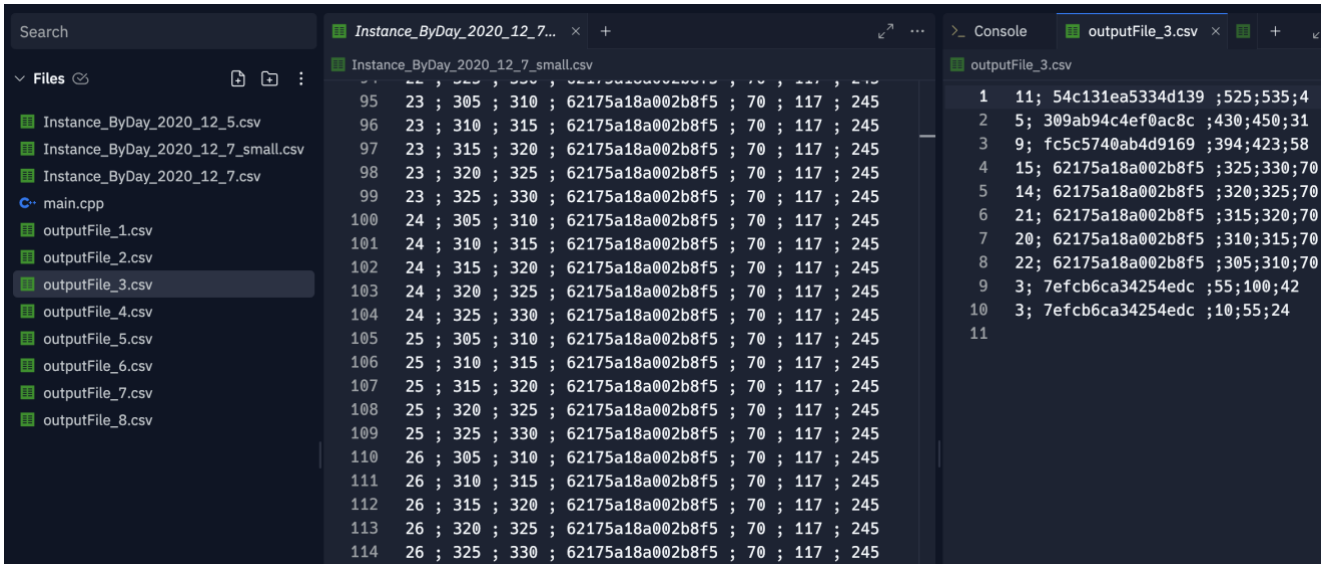


Figure 8: original csv and output file(s)

- **Optimization:**

  Through We need to “correct” the algorithm

  - We Save the list of beginning (O)
  - Save the list of intervals (R)
  - Save the list of remaining intervals (S)
  - R = O – S

Then, the last thing that we will have to do is to remove intervals that have the same hash code. Because identical hash code means identical TV shows. We don't need to record it twice.

Number of cards	Number of reruns	Number of selected programs	Total weight after selection
4	1	19	95
	2	14	81
	3	10	71
	4	8	63
	total	51	310
6	1	19	95
	2	14	81
	3	10	71
	4	8	63
	5	9	54
	6	7	47
	total	67	411
8	1	19	95
	2	14	81
	3	10	71
	4	8	63
	5	9	54
	6	7	47
	7	7	40
	8	6	34
	total	80	485

Figure 9: The Kleinberg algorithm with different parameters

we can create a checker to verify that int output files an interval does not appears twice times.we can create also function check_overlap. This function takes only1 output file.A second function, that open several files. That function can open the output_*. csv file, and read them and check that an interval does not appears twice times.

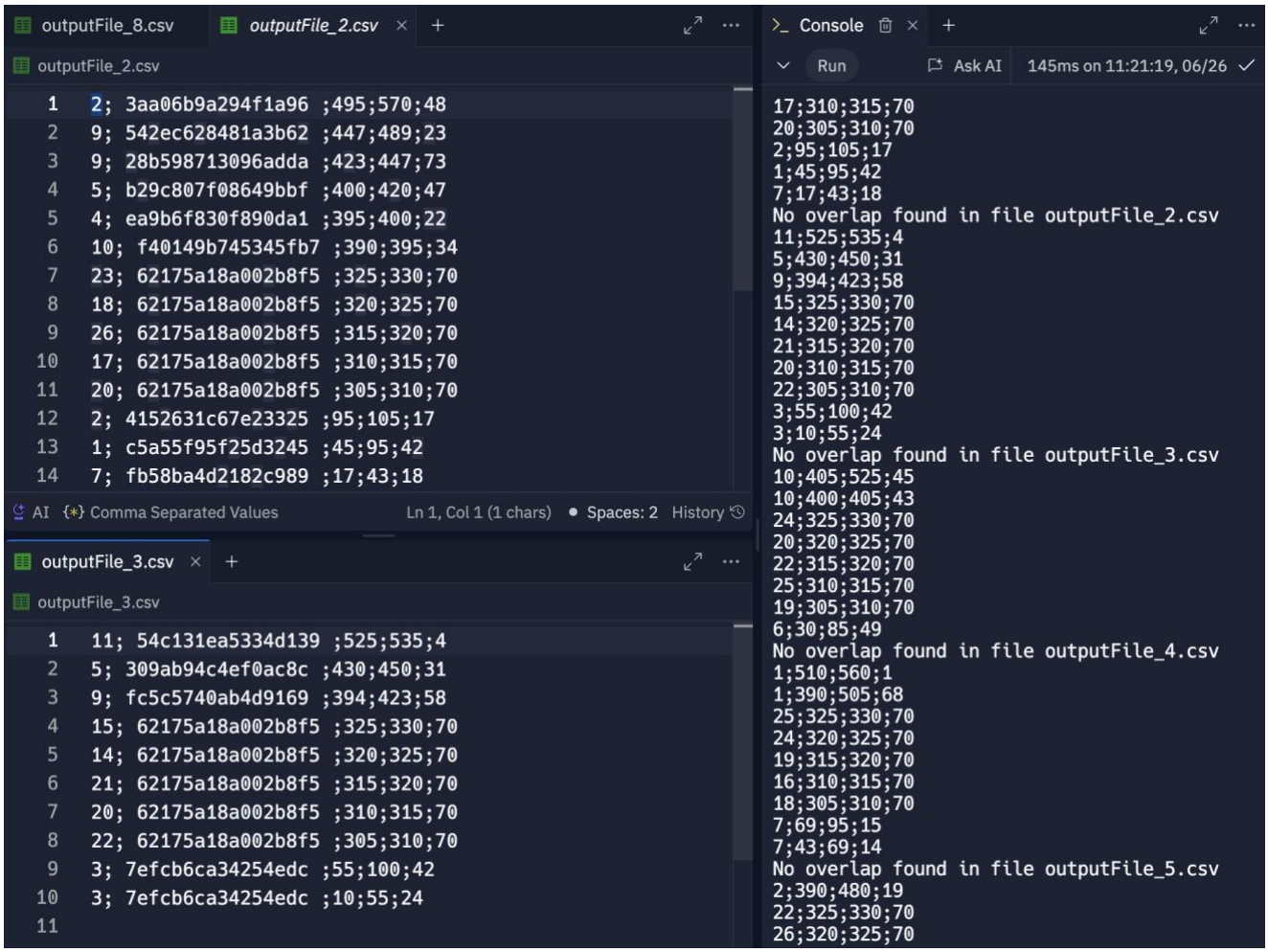


Figure 10: Check for overlapping results.

Through this iterative approach, we ensure efficient utilization of all available resources and optimize the recording process across multiple machines.

- **Remarks:**

In this issue, when two time slots have the same hash code, it means they represent the same program. Ideally, it is best to avoid recording the same program multiple times. However, unlike the Greedy algorithm, Kleinberg's algorithm was not initially designed to address this issue. Therefore, additional processing needs to be integrated into the algorithm to ensure that two time slots with the same hash code are not selected simultaneously. This improvement will help optimize the algorithm's performance by preventing duplicate recordings and enhancing overall efficiency in program selection.

4. The runtime for dynamic video capture

4.1.Program Stucture.

In this project, the program starts by reading the TV broadcast schedule from a CSV file. This file contains detailed information about the intervals of TV channel broadcasts, including start time, end time, program hash code, and the weight of each time slot. This helps in collecting the necessary data for processing and scheduling automatic channel changes..

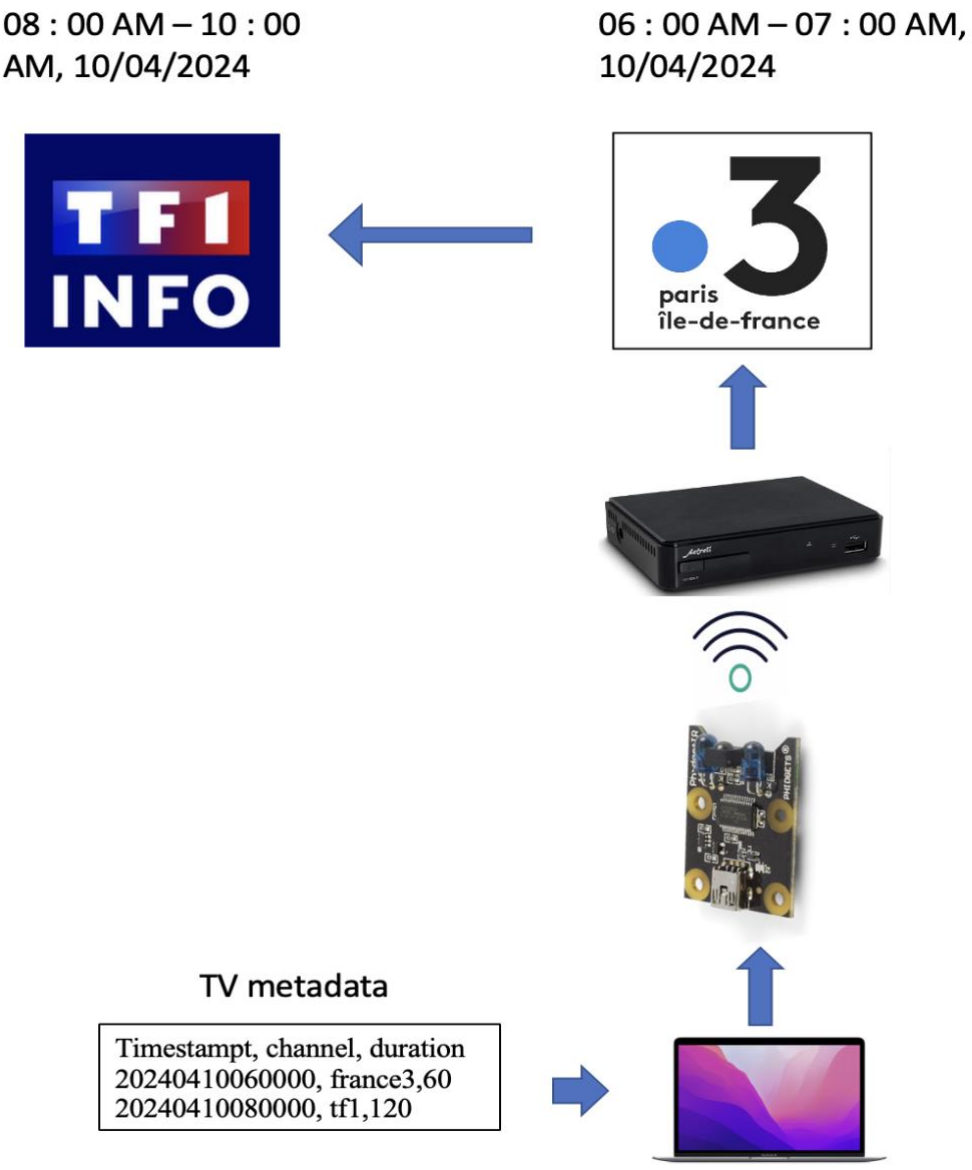


Figure 11: Program's cycle

A program for switching TV channels and setting the channel change time (T) has been developed. This program uses Kleinberg's Interval Selection Algorithm to select the optimal time slots from the read schedule. After selecting the optimal time slots, the program will set the channel change time to best fit the schedule..

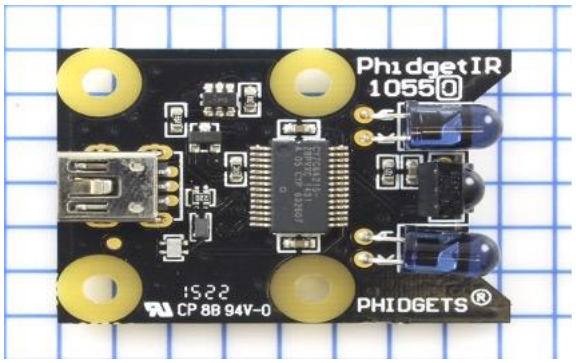
The program will send the channel transmission code from the sensor to the tuner at the scheduled time. This transmission code is used to inform the tuner which channel needs to be broadcast at the specific time. Sending this code is done automatically, saving time and effort compared to manual channel switching.

Finally, the program will automatically switch TV channels according to the optimal schedule. This ensures that viewers do not miss any of their favorite programs and optimizes their TV viewing experience. The automatic channel switching program enhances the efficient use of video cards and ensures that channels are switched smoothly and exactly according to the planned schedule.

4.2.Control phidgets sensor.

This Phidget device enables the transmission and reception of IR data, allowing users to learn and replicate signals from commonly used devices such as TV remotes.

First, the program uses the Phidget22 library. The Phidget22 library is a software toolkit designed to interact with Phidget devices. Phidget devices are versatile sensors and controllers that can be connected to a computer via USB or network. The Phidget22 library provides a programming interface (API) to program and control Phidget devices from various programming languages such as Python, C#, C++, Java, etc.



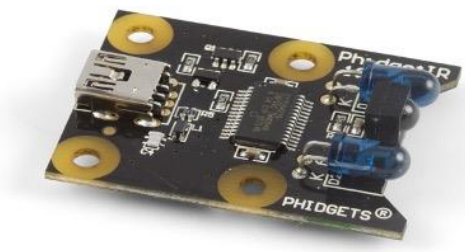


Figure 12.1: Phidget sensor.

Figure 12.2: Hardware mounting kit

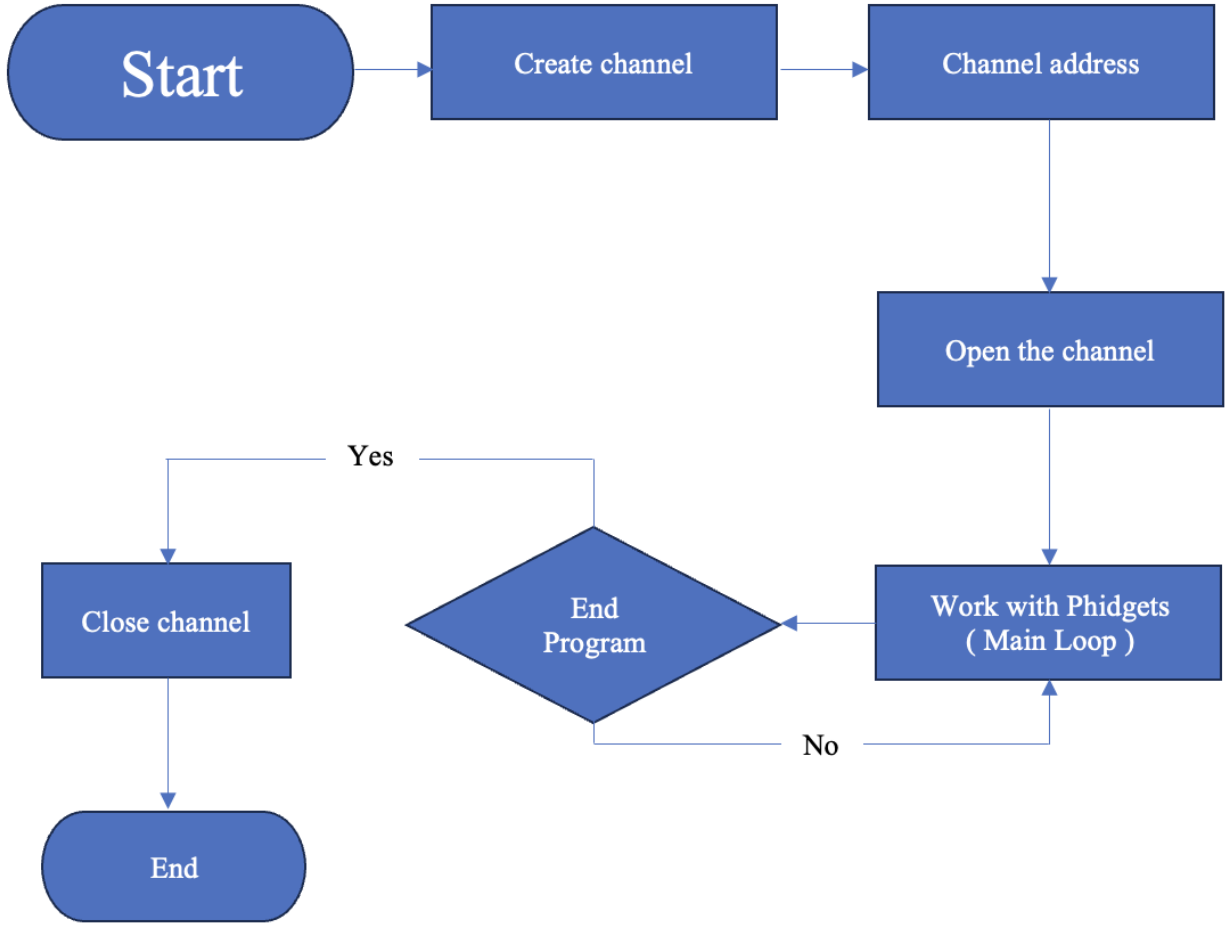
Figure 12.3: USB Mini-B.

To program with Phidgets, very important to understand how the physical input and output channels of Phidget devices correspond to the software controlling them.

Each part of Phidget's behavior is divided into individual parts called Channels. Each channel represents a piece of equipmen. Each channel is associated with a Channel Class, indicating the specific function performed by the channel.

Each channel class is designed to make using channels of that class as similar as possible, regardless of the Phidget they are a part of.

Second, Phidgets program, any channel Phidget uses in any program will follow the same cycle as follows:



1. **Create a channel**: Make a variable to keep track of your Phidget

2. **Address the channel:** Set some basic parameters to indicate which Phidget to connect to.
   - We can specify as much, or as little as you deem necessary. We can even use a Phidget without specifying any parameters, if our system is simple enough.

3. **Open the channel**: Opening the channel will begin trying to match a physical Phidget channel to our software channel.

4. **Detect when the channel is attached:** Our program must wait until a physical Phidget channel has been attached to our software channel.
   - The attachment process is handled automatically, and will attach to the first available

     Phidget channel that matches all our specified addressing parameters.
   - A channel is attached when the Phidget libraries link a physical Phidget channel to our software channel.
   - Waiting for attachment can be handled automatically as part of opening the channel or separately, as best suits our program.

5. **Do things with the channel:** Send commands to and receive data from your Phidget.
   - This section will encompass the majority of your program. The Phidget API / libraries will make this process as straightforward as possible, and is similar across all types of Phidgets.

6. **Close the channel:** Once your program is done, you close the Phidget channel.
   - This frees up the Phidget for the next program that might need it.

For a closer look at the parts of a Phidget program, the following pages will dissect the parts of a Phidget program in more detail. Small code snippets are provided for each step to provide a representation of the code.

Third, to control the IR Phidgets sensor, Library 22 is used to transmit IR codes. These codes consist of 32-bit data. The program utilizes a method called IR_ENCODING_SPACE, where the timing between pulses determines the value of each bit (0 or 1).

1. `codeInfo = CodeInfo()`

- Description: Initializes a `CodeInfo` object, which is a data structure used to store the necessary encoding parameters for transmitting IR codes.

- Purpose: Creates a `CodeInfo` object to configure IR code transmission parameters.

2. `codeInfo.bitCount = 32`

- Description: Sets the number of bits in the IR code to 32.

- Purpose: The IR code will consist of 32 bits of data.

3. `codeInfo.encoding = IRCodeEncoding.IR_ENCODING_SPACE`**

- Description: Sets the encoding method for the IR code to `IR_ENCODING_SPACE`.

- Purpose: `IR_ENCODING_SPACE` is an encoding method where the timing between pulses determines the bit values (0 or 1). It's a common IR encoding type.

4. `codeInfo.zero = [580, 580]`

- Description: Sets the pulse duration for bit 0 to [580, 580] microseconds.

- Purpose: Bit 0 is encoded using two pulses, each lasting 580 microseconds.

5. `codeInfo.one = [580, 1710]`

- Description: Sets the pulse duration for bit 1 to [580, 1710] microseconds.

- Purpose: Bit 1 is encoded using two pulses: the first pulse lasts 580 microseconds and the second pulse lasts 1710 microseconds.

6. `codeInfo.header = [9080, 4550]`

- Description: Sets the header pulse to [9080, 4550] microseconds.

- Purpose: Before transmitting data, a header pulse is sent to indicate the start of the IR code, consisting of a pulse of 9080 microseconds followed by a pulse of 4550 microseconds.

7. `codeInfo.trail = 580`

- Description: Sets the trailer pulse to 580 microseconds.

- Purpose: A 580-microsecond pulse is sent after the entire data transmission to signal the end of the IR code.

8. `codeInfo.gap = 108500`

- Description: Sets the gap between IR codes to 108500 microseconds.

- Purpose: The time interval between consecutive IR codes is 108500 microseconds. This is the waiting period before a new IR code can be transmitted.

This code configures specific parameters for IR code transmission.These parameters include the number of bits, encoding method, pulse durations for bit 0 and bit 1, header and trailer pulses, and the gap between IR codes. All of these configurations define how the IR code will be transmitted, ensuring that receiving devices (like TVs) can process the signal correctly.

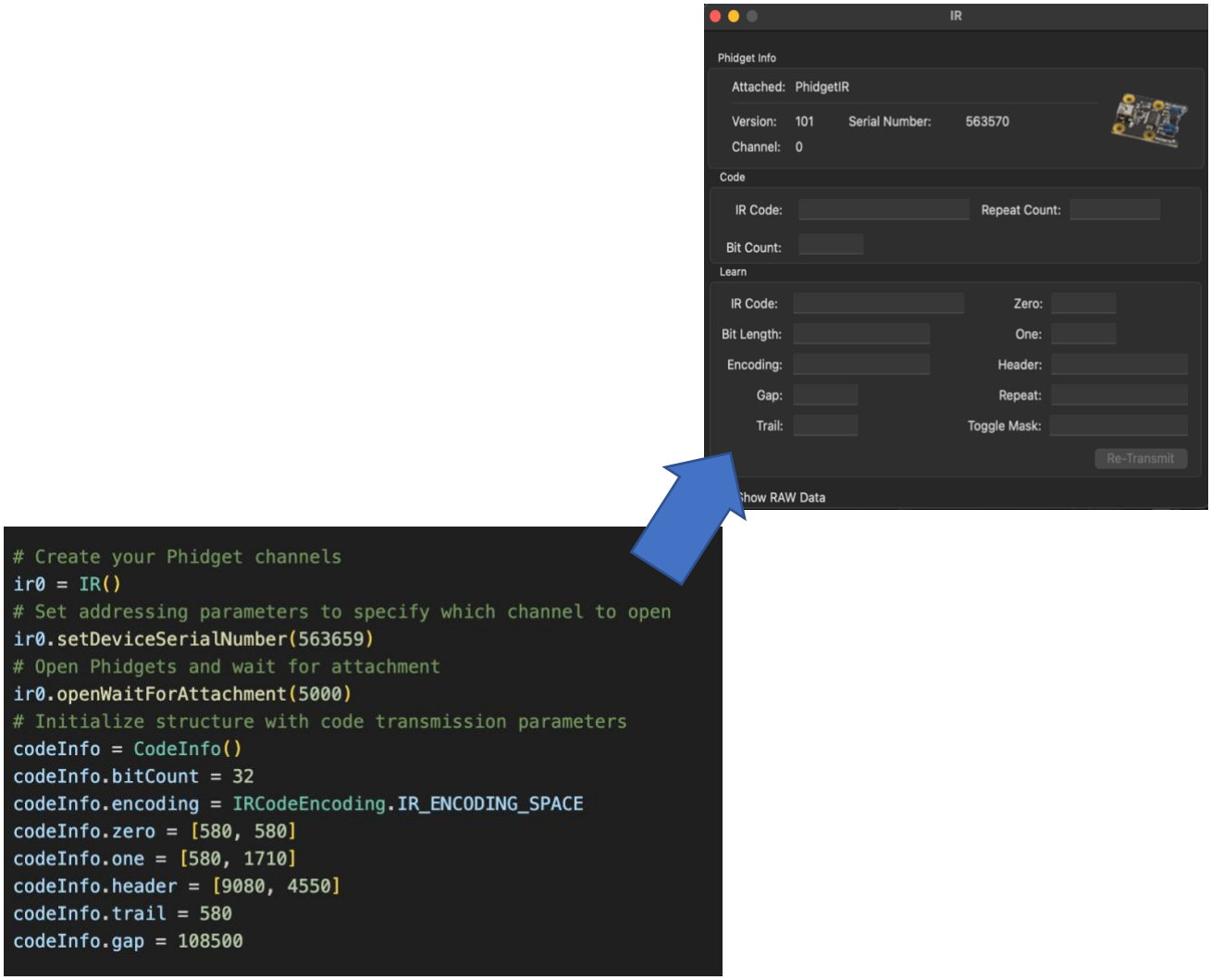


Figure 3:IR_ENCODING_SPACE.

4.3.Channel switching time.

As mentioned in section 4.1, the program begins by reading the TV broadcast schedule from a CSV file. This file contains detailed information about the intervals of TV channel broadcasts, including start time, end time, program hash code, and the weight of each interval. Therefore, it's necessary to read data from the CSV file and convert the 'start_time' and 'stop_time' columns into Python's time format, to prepare for processing and analyzing time intervals in a pandas DataFrame.

```
# Read the CSV file
df = pd.read_csv(r"/Users/lexuanquang/Documents/Python_demo/tv-programs.csv")
# Convert 'start_time' and 'stop_time' columns to datetime with format string
df['start_time'] = pd.to_datetime(df['start_time'], format='%H%M%S').dt.time
df['stop_time'] = pd.to_datetime(df['stop_time'], format='%H%M%S').dt.time
```

To optimize automatic channel switching, the program needs to calculate the appropriate channel switching time based on the TV schedule data. Here's how it can be structured:

index	**channel_name**	**start_time**	**stop_time**
0	Channel 2	18:58:00	18:59:00
1	Channel 3	19:01:00	19:02:30
2	Channel 5	19:04:45	19:05:00

- An example csv table like above, It follows the structure :

ch1; t0; t1
ch2; t2; t3
ch3; t4; t5
....

where t0 < t1 < t2 .... < t7 and ch1 * ch2 …..*ch4 (in a simple way, ch2 = ch1 +1)

➢ Set channel switching time:

$$T = t1 + \frac{(t2 - t1)}{2}$$

- T: Switch time
- t1 is the stop_time of the current channel
- t2 is the start_time of the next channel

Exception: The first channel, the channel switch time will be 1 minute before the start time. Because with the first channel, we will not have data about the end time of the previous channel. Therefore we can choose the time to start switching channels.

```python
def my_task():
    while True:
        now = datetime.now().replace(microsecond=0)
        current_time = now.strftime('%H:%M:%S')
        # Sắp xếp DataFrame theo start_time / Sort DataFrame by start_time
        sorted_df = df.sort_values(by='start_time')

        for i in range(len(sorted_df) - 1):
            ''' Vòng lặp này sẽ lặp qua các hàng của DataFrame sorted_df,
            ngoại trừ hàng cuối cùng. Điều này là vì khi chúng ta đang xử lý hàng hiện tại row_current,
            chúng ta cần truy cập hàng tiếp theo row_next ở vị trí i + 1. '''
            row_current = sorted_df.iloc[i]
            row_next = sorted_df.iloc[i + 1]

            # Kiểm tra nếu là kênh đầu tiên/Check if the channel is first
            if i == 0:
                start_time_first_channel = datetime.combine(datetime.today(), row_current.start_time)
                default_switch_time = start_time_first_channel
                if now == default_switch_time.replace(microsecond=0):
                    print("Time:", current_time)
                    print(f"Switching to channel {row_current.channel_name} at switch time {default_switch_time}")
                    switch_channel(row_current.channel_name)
                    break

            stop_time_current = datetime.combine(datetime.today(), row_current.stop_time)
            start_time_next = datetime.combine(datetime.today(), row_next.start_time)

            switch_time = stop_time_current + (start_time_next - stop_time_current) / 2

            if now == switch_time.replace(microsecond=0):
                print("Time:", current_time)
                print(f"Currently capturing channel: {row_next.channel_name} at switch_time {switch_time}")
                switch_channel(row_next.channel_name)

        time.sleep(1)  # Đợi 1 giây trước khi kiểm tra lại
```

To develop, The `my_task()` function is designed to automate channel switching on a TV based on time information from the `tv-programs.csv` file. Here's an explanation of how channel switching time is handled in the code:

1. Get current time (`now`):

- The line `now = datetime.now().replace(microsecond=0)` retrieves the current time and removes microseconds for easier comparison with times in the DataFrame.

2. Sort DataFrame by `start_time` (`sorted_df`):

- The line `sorted_df = df.sort_values(by='start_time')` sorts the DataFrame `df` by the start time (`start_time`) of each channel, from earliest to latest.

3. Iterate through DataFrame rows (`for i in range(len(sorted_df) - 1)):` :

- This loop iterates over each row of the sorted DataFrame `sorted_df`, excluding the last row, to compare the current time with the start and end times of each channel.

4. Handling the first channel (`if i == 0):`:

- If `i == 0`, it means we are handling the first channel in the sorted list.

- Calculate `default_switch_time`, which is the default time to switch to the first channel. This time is computed by adding the start time of the first channel to the current date and time.

- If the current time matches `default_switch_time`, print information about the current time and channel, and execute the channel switch using the `switch_channel()` function.

5. Handling subsequent channels:

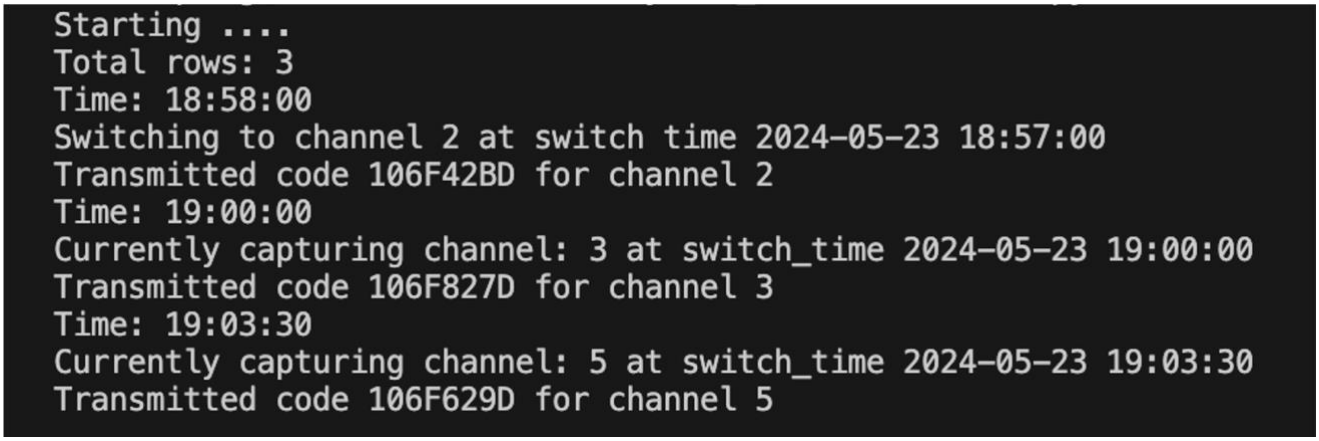
- For next channels (`i > 0`), calculate `switch_time`, which is the time to switch from the current channel to the next one. This time is calculated by taking the end time of the current channel and adding half of the time gap between the end time of the current channel and the start time of the next channel.

- If the current time matches `switch_time`, print information about the current time and the next channel, and execute the channel switch using the `switch_channel()` function.

6. Wait for 1 second (`time.sleep(1)`) before checking again:

- To reduce CPU load and limit frequent time checks, the program pauses for 1 second before repeating the time check loop.

Results when running in real time:



5. Conclusions and perspectives

This code configures specific parameters for IR code transmission.These parameters include the number of bits, encoding method, pulse durations for bit 0 and bit 1, header and trailer pulses, and the gap between IR codes. All of these configurations define how the IR code will be transmitted, ensuring that receiving devices (like TVs) can process the signal correctly.

During the process of recording multiple channels, storage requires a lot of resources and is prone to duplicate data issues. To address this problem, we developed a two-level solution.

First, we developed a program to automatically switch channels. This program helps switch between channels automatically based on a preset schedule.

Second, we applied Kleinberg's algorithm to optimize the recording schedule. This algorithm helps select the most valuable recording intervals, avoiding duplicate recordings.

As a result, we have created an automated and optimized system for efficiently recording TV programs.

- **Future Work**

- Continue to improve the performance of Kleinberg's algorithm to enhance optimization capabilities.
- Develop a more stable video recording program with smooth and accurate automatic channel switching.

- **Acknowledgement**

We would like to express our sincere thanks to lecturer Tifenn Rault and professor Mathieu Delalandre. The guidance and dedicated support of our teachers helped us complete this project.

Thanks to Professor Mathieu's detailed instructions and valuable advice, I had the opportunity to learn and develop many important skills. The enthusiastic support and encouragement of lecturer

Tifenn helped me overcome difficulties and become more passionate about algorithms throughout the project.

Once again, we would like to sincerely thank you for everything you have done. We are very grateful and appreciate your help and guidance during the past internship period.

Supplement

---------- 2024xxxx ----------

Following the return from the trip to Vietnam, a follow-up on the Pal ticket no. 84080 was initiated on 04/06 concerning the 20 TB #HDD drives. The drives were received on 21/05, the delivery note was retrieved, and the invoice was paid (posted online, see tvstation-ldlc). The inventory labels were collected on 04/06 from V. Lasnier. The two drives were retrieved on 06/06, following their registration/inventory at UT. They were labeled as HDD08 and HDD09 and recorded (see servers-backup file).

---------- 20240515 ----------

Following discussions with N. Ragot and F. Rayar during the weeks of 08/04 and 15/04, budget agreements were formulated for the purchase of 20 TB #HDD storage drives for the TV station: (i) 2x 20 TB drives from the DI investment budget and (ii) 1x 20 TB drive from the ART budget. It was decided to freeze the request on the ART budget (ii) and proceed with the DI investment budget (i). It was noted, after discussions with N. Ragot and V. Lasnier during the weeks of 08/04 and 22/04, that the purchase had been approved by the department but would require a presentation to the BR committee (i.e., budget rectification) beforehand (to request an extension of the department's investment budget, which was exhausted for the 2023-2024 fiscal year). This request will be discussed in June (for a purchase commitment in September, if funded). For now, only a quotation is needed before presenting it to the BR.

An email from V. Lasnier was sent on 18/04 requesting a quotation for two large-capacity LaCie d2 Professional USB 3.1 20 TB drives [1]. A Pal ticket no. 84080 was opened on the same day. The drives are expected to cover a budget of €1.48k including VAT (€615*2 excluding VAT) and can be purchased from the investment budget (rule >€500 excluding VAT per unit). An email was sent (M. Delalandre -> V. Lasnier, S. Beaufils) on 18/04 to initiate the quotation request. A follow-up was initiated on 22/04. After discussion with S. Beaufils on 22/04, a direct exchange procedure with the supplier was proposed as the simplest solution (administration@ldlc.pro / Client no. E37ECOLEXW000). The quotation was received on 22/04 (in two phases after correcting the quotation

from x1 to x2 drives due to an error by LDLC) for the amount of €1.48k (posted online, see tvstation-ldlc). LDLC was informed of the order process (i.e., in BR) in response. After updating the Pal ticket no. 84080 on 22 and 23/04, it was frozen/deferred in anticipation of the BR return and order during the June to September period.

After a discussion with V. Lasnier on 24/04, it appears that canceling a purchase on the DAE investment (approximately €5k) would allow the request to proceed asap (without going through the BR). The Pal ticket no. 84080 was updated on 24/04 with this information to cancel the deferral. The purchase order was issued, printed, and sent for signature on 25 and 29/04, signed, and sent to LDLC to confirm the order on 16/05 (also posted online, see tvstation-ldlc).

Emails were sent (M. Delalandre -> F. Rayar and M. Delalandre -> N. Ragot) to inform about the request and the freezing of ART/preciput funding (on 23 and 24/04).

[1] LaCie d2 Professional USB 3.1 20 TB https://www.ldlc.pro/fiche/PB00502054.html ¶

[1] F. Rayar, M. Delalandre and V.H. Le. A large-scale TV video and metadata database for French political content analysis and fact-checking. Conference on Content-Based Multimedia Indexing (CBMI), pp. 181-185, 2022.

[2] Large-Scale TV Dataset Parallel Machine Scheduling (STVD-PMS) https://dataset-stvd.univ-tours.fr/pms/

[3] T. Erlebach, F.C.R. Spieksma. Interval selection: Applications, algorithms, and lower bounds. Journal of Algorithms, vol 46(1), pp. 27-53, 2003.

[4] C. Lezé–Robert. Optimisation pour la sélection de programmes TV. End-year student project, Polytech Tours institute, 2023.

[5] A. Lomaszewicz and T. Mercier. Evaluation de performance de l'algorithme GREEDYα sur la base de données STVD-PMS. End-year student project, Polytech Tours institute, 2024.

[6] V.H. Le, M. Delalandre and D. Conte. A large-Scale TV Dataset for partial video copy detection. International Conference on Image Analysis and Processing (ICIAP), Lecture Notes in Computer Science (LNCS), vol 13233, pp. 388-399, 2022.

[7] M. Doyen and A. Balois. Detection smart de latence sur station TV. End-year student project, Polytech Tours institute, 2023.

Sources

To get the Phidgets 22 library for your operating system, install it from

https://www.phidgets.com/

In our problem, we placed an order for USB Type A A and USB Mini-B cables through

https://www.ldlc.pro/

Additionally, we procured a Remote Control PhidgetIR from

https://www.phidgets.com/ .

Large-Scale TV Dataset Parallel Machine Scheduling (STVD-PMS):
https://dataset-stvd.univ-tours.fr/pms/

The TV Workstation project: a research scope

http://mt.free.fr/

A Workstation for Real-Time Processing of Multi-Channel TV

https://hal.science/hal-03552616

A large-scale TV video and metadata database for French political content analysis and fact-checking

http://mathieu.delalandre.free.fr

Real-time detection of partial video copy on TV workstation

http://mathieu.delalandre.free.fr/